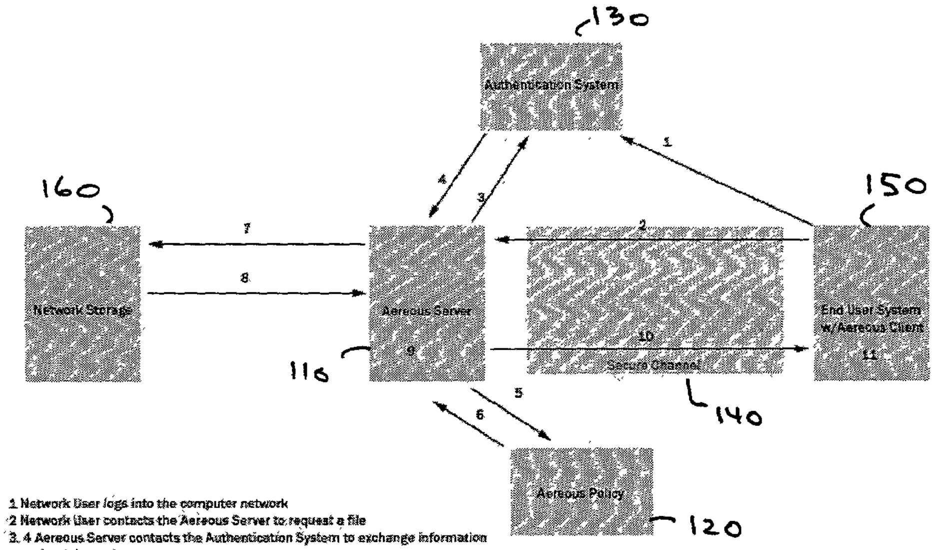


1 Network User logs into the computer network
2 Network User contacts the Aereous Server to request a file
3, 4 Aereous Server contacts the Authentication System to exchange information about the end user
5, 6 Aereous Server contacts the Aereous Policy System and executes the access policy to determin if the user has the priviliages to access the file. If the answer is yes. then the usage policy is sent to the Aereous Server, if no then the file is not sent to the end user
7 Aereous Server requests the file from the network storage device
8 Network storage device delivers the file to the Aereous Server
9 Aereous Server applies usage rights to, and encrypts. the file
10 The file is securely delivered to the End User System
11 Usage rights and auditing is enforced on the End User System by the Aereous Client

FIG. 1

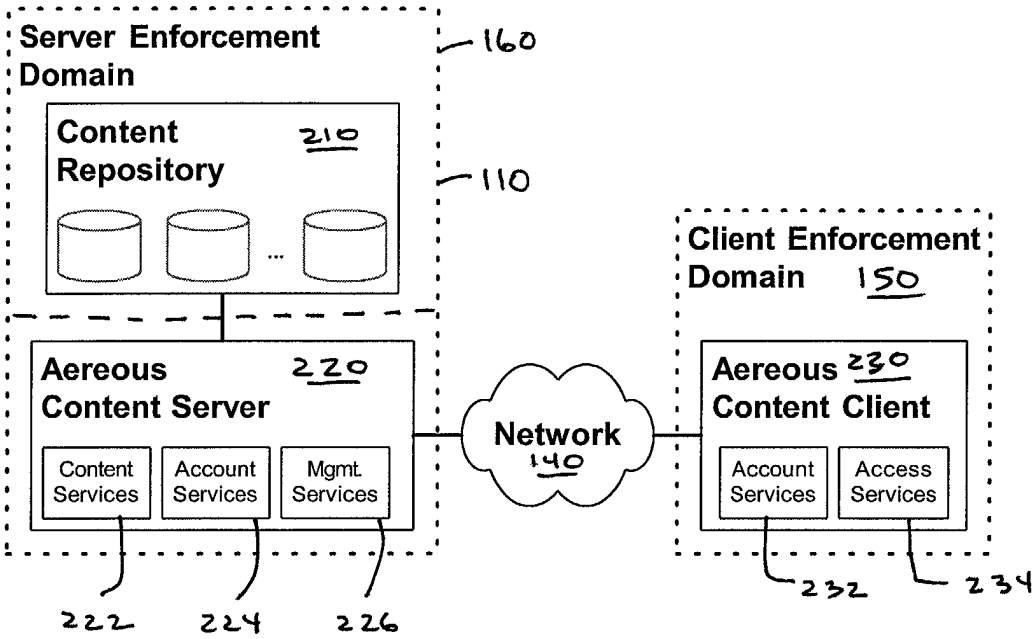

Server Enforcement Domain
160
200
Content Repository
210
110
Client Enforcement Domain
150
Aereous Content Server
220
Aereous Content Client
230
Content Services
Account Services
Mgmt. Services
Network
140
Account Services
Access Services
222
224
226
232
234

FIG. 2

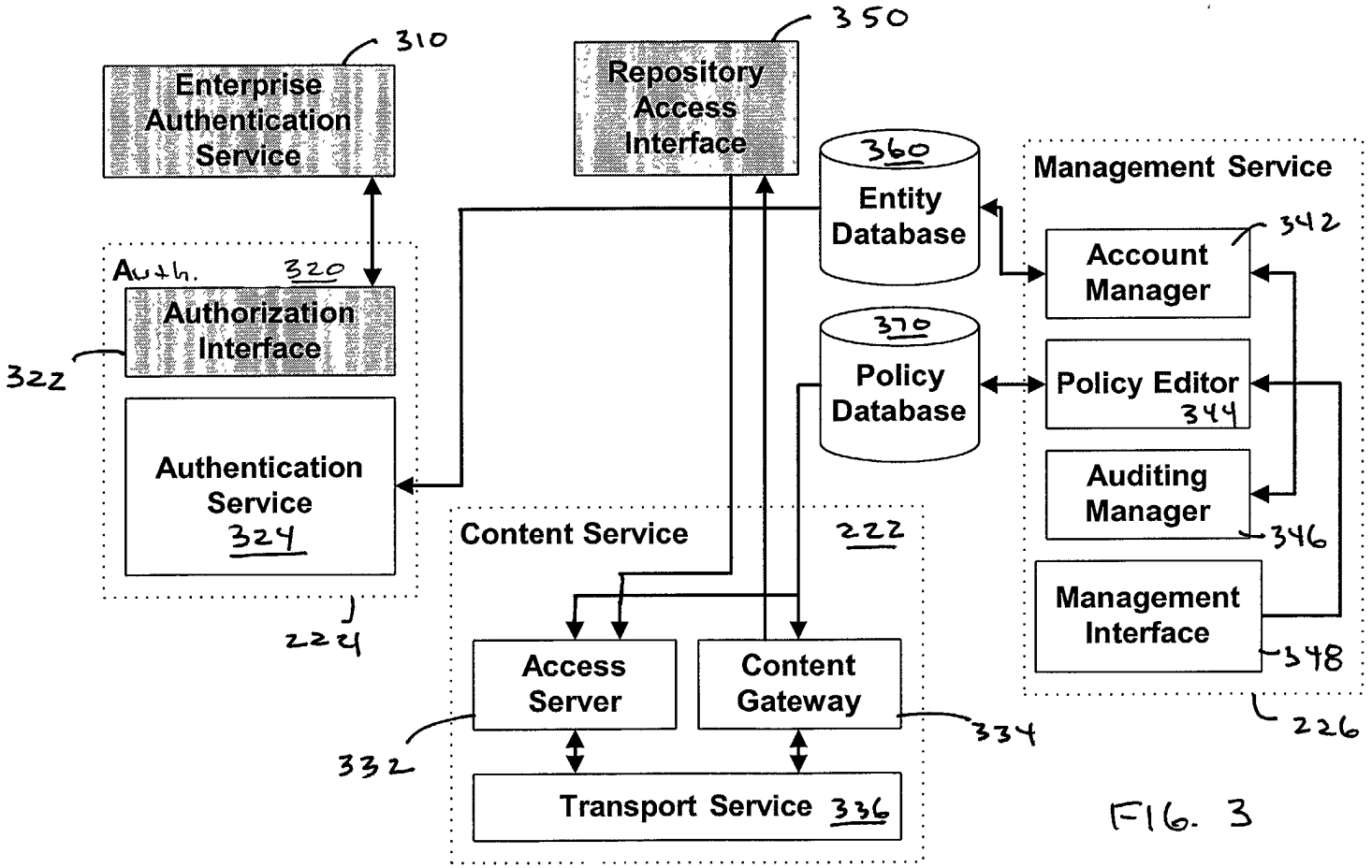


FIG. 3

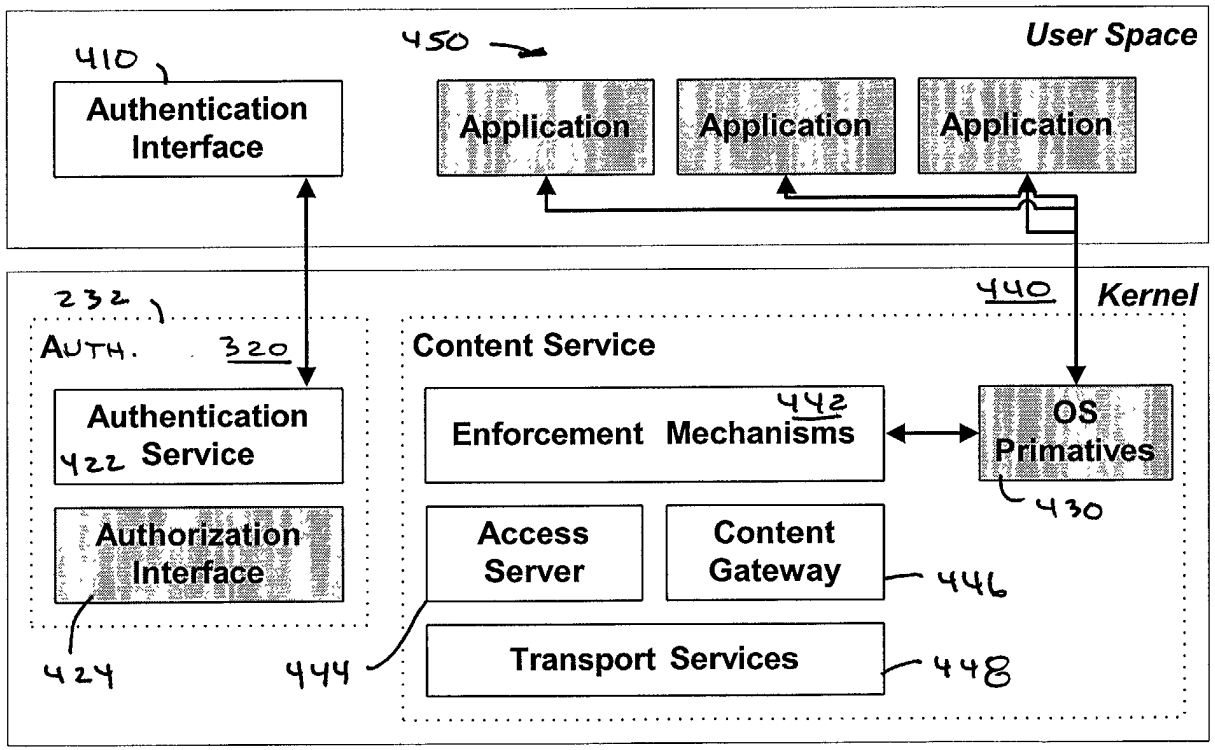

User Space
450
410
Authentication Interface
Application
Application
Application
232
440
Kernel
AUTH.
320
Content Service
Authentication Service
422
442
Enforcement Mechanisms
OS Primatives
430
Authorization Interface
Access Server
Content Gateway
446
424
444
Transport Services
448

FIG. 4

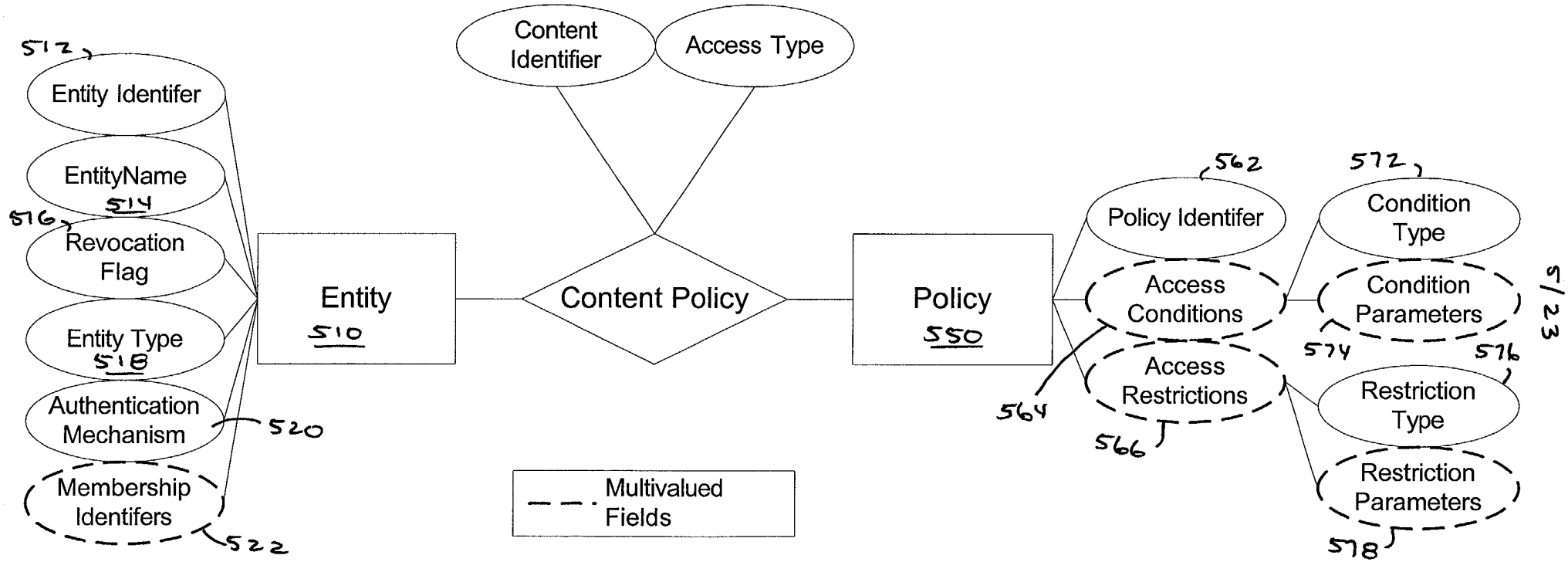

512
Entity Identifer
EntityName
514
516
Revocation Flag
Entity Type
518
Authentication Mechanism
520
Membership Identifers
522
Entity
510
Content Identifier
Access Type
Content Policy
Multivalued Fields
Policy
550
562
Policy Identifer
Access Conditions
564
Access Restrictions
566
572
Condition Type
Condition Parameters
574
576
Restriction Type
Restriction Parameters
578

FIG. 5

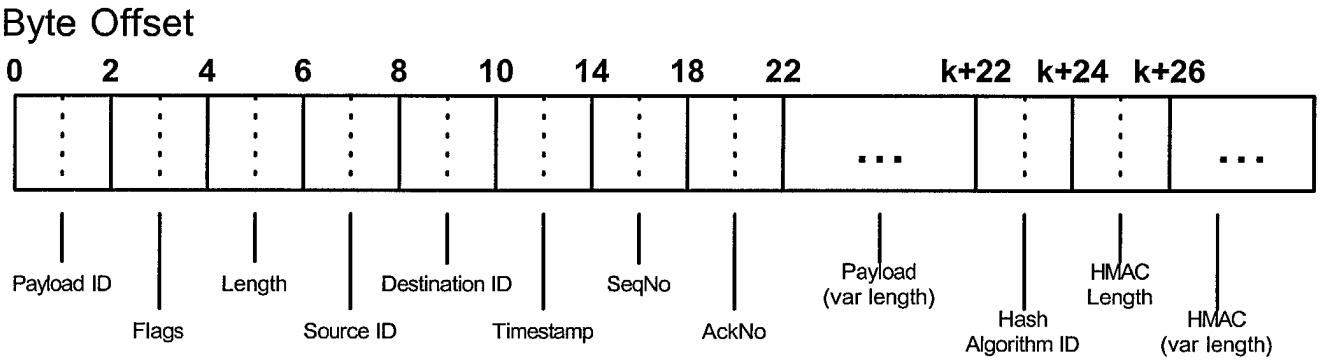

Byte Offset
0
2
4
6
8
10
14
18
22
k+22
k+24
k+26
...
...
Payload ID
Flags
Length
Source ID
Destination ID
Timestamp
SeqNo
AckNo
Payload
(var length)
Hash
Algorithm ID
HMAC
Length
HMAC
(var length)

FIG. 6A

Field	Length	Description
Payload ID	2 bytes	Enumerated type describing the payload type of the message: Further processing of the message is directed by this field. The currently payload identifiers include **Type** **Value** **Description** AERE_INVALID 0 Invalid type AERE_FILE_KEY 1 File key payload AERE_BLK_XFER 2 Block transfer AERE_STATUS 3 Aereous status
Flags	2 bytes	Flags indicating payload processing requirements. The currently defined flags include; **Flag** **Bit** **Description** Encrypted 0 Payload encrypted Signed 1 Payload signed *(not implemented)* Reserved 2-15 *unused*
Length	2 bytes	Length of message, in bytes. This length measures the field through the the last byte of the payload.
Source ID[2]	2 bytes	Source identifier - uses user or server entity identifier defined in the entity database.
Destination ID	2 bytes	Recipient identifier - uses user or server entity identifier defined in the entity database.
Timestamp	4 bytes	Timestamp (obtained from local or trusted timing source) of message creation. Used to ensure freshness (e.g., mitigate replay attacks). The time is represented by the standard POSIX 32 bit second identifier (seconds since epoch)..
SeqNo	2 bytes	Sequence number used to ensure the ordering of messages.
AckNo	2 bytes	Acknowledgement of all messages up to including *Ackno*.
Payload	*variable*	This is the variable length data to be interpreted by payload processing. The format of the payload is detailed in Section 7.3. Based on message flags, this data require additional proces (e.g., encryption, sign).
Hash Algo. Identifier	2 bytes	Enumerate type defining the hash algorithm used in the calculation of the keyed hash. The following hash algorithms are supported by the Aereous system; **Algorithm** **Value** AERE_MD5 0 AERE_SHA1 1
HMAC Length	2 bytes	The length of the HMAC value. Note that some crypgraphic algorthms output more ciphertext than the orginal plaintext. (Question; Is this really needed, or can we always calculate this from the key/hash algorithm info?)
HMAC	*variable*	This is the keyed hash of the message. This value is calculated over all byes prior to the begining of the hash length field.

FIG. 6B

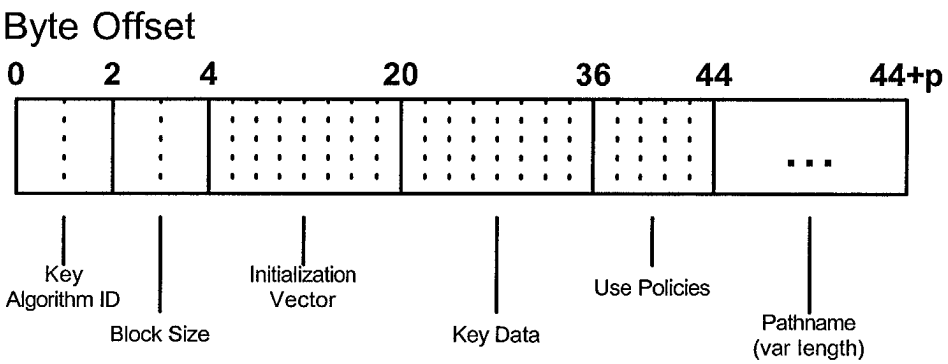


FIG. 7

Name	Length	Description
KeyAlgorithmID	16 bits	(enumerated) identifies both the algorithm and the key length
BlockSize	16 bits	block size for the accessed file
IV	256 bits	Intialization vector used to seed the encryption of file blocks. Further details are defined in Section 7.1.
KeyData	256 bits	The key used to encrypt the file. Where the key size is less than 256 bits, the most significant bits are used and unused bits are padded with zero.
UsePolicies	64 bits	Flags indicating the enabled usage of accessed content (where a bit 1=allowed, 0=denied). The supported bits include; Flag / Bit / Description Print / 0 / Print the file Copy / 1 / Copy file to local disk Send / 2 / Transmit the file to external device Reserved / 3-63 / *unused* *NOTE: The set of usage types are identified in the Aereous Client Design Document, and will be reflected in future version this document as needed.*
Pathname	(variable)	full pathname of file being accessed

FIG. 8

Name	Length	Description
Cid	16 bits	hashed pathname identifier (see Section 6)
BlockNumber	16 bits	block number of transmitted data
Length	16 bits	length of data. Typically equal to the block size supported by the filesysem.
Data	(variable)	the file data

FIG. 9

Name	Length	Description
Sid	16 bits	(enumerated) Type identifying the message semantics. *Details of the status are further specified in the info and text fields.* Enum / Numeric / Origin / Description usageExec / 0 / client / Usage right executed aereousError / 1 / both / Aereous error encountered dfsError / 2 / both / Filesystem error infoStatus / 3 / both / informational (e.g., debugging) clientShutdown / 4 / client / client shutdown signal serverShutdown / 5 / server / server shutdown signal *unused* / 6-2^16-1 / N/A / *unused*
InfoLength	16 bits	length of *info* field.
Info	*(variable)*	Additional status information. The interpretation of this field is directed by the *Sid* field as follows: Enum / Subfields usageExec / content ID (*cid*), usage mask aereousError / Aereous error code dfsError / standard UNIX errno infoStatus / information enum clientShutdown / *none* serverShutdown / *none* *unused* / *unused*
TextLength	16 bits	length of *Text* field.
Text	*(variable)*	C-string description of information. Used in auditing or as user notification.

FIG. 10

Figure 11

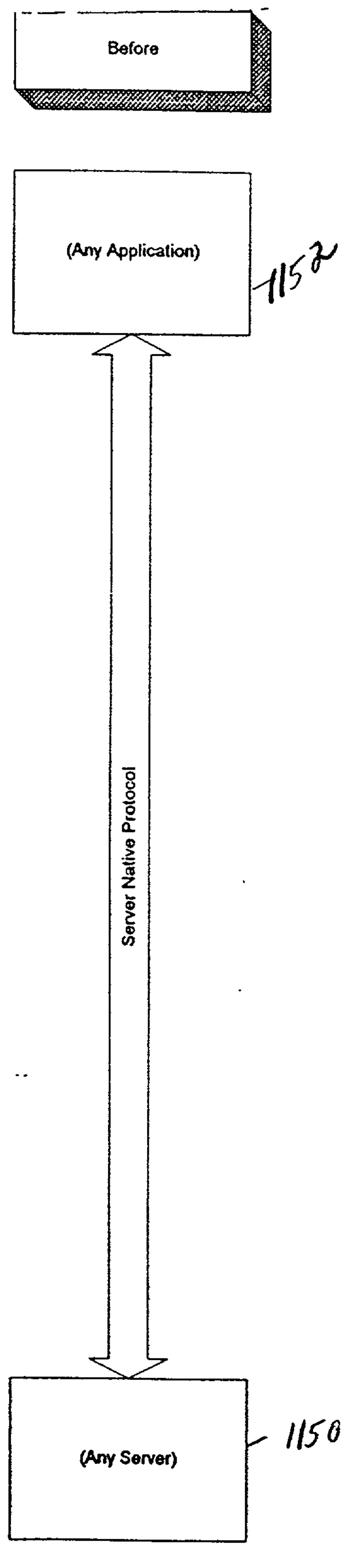


Figure 12

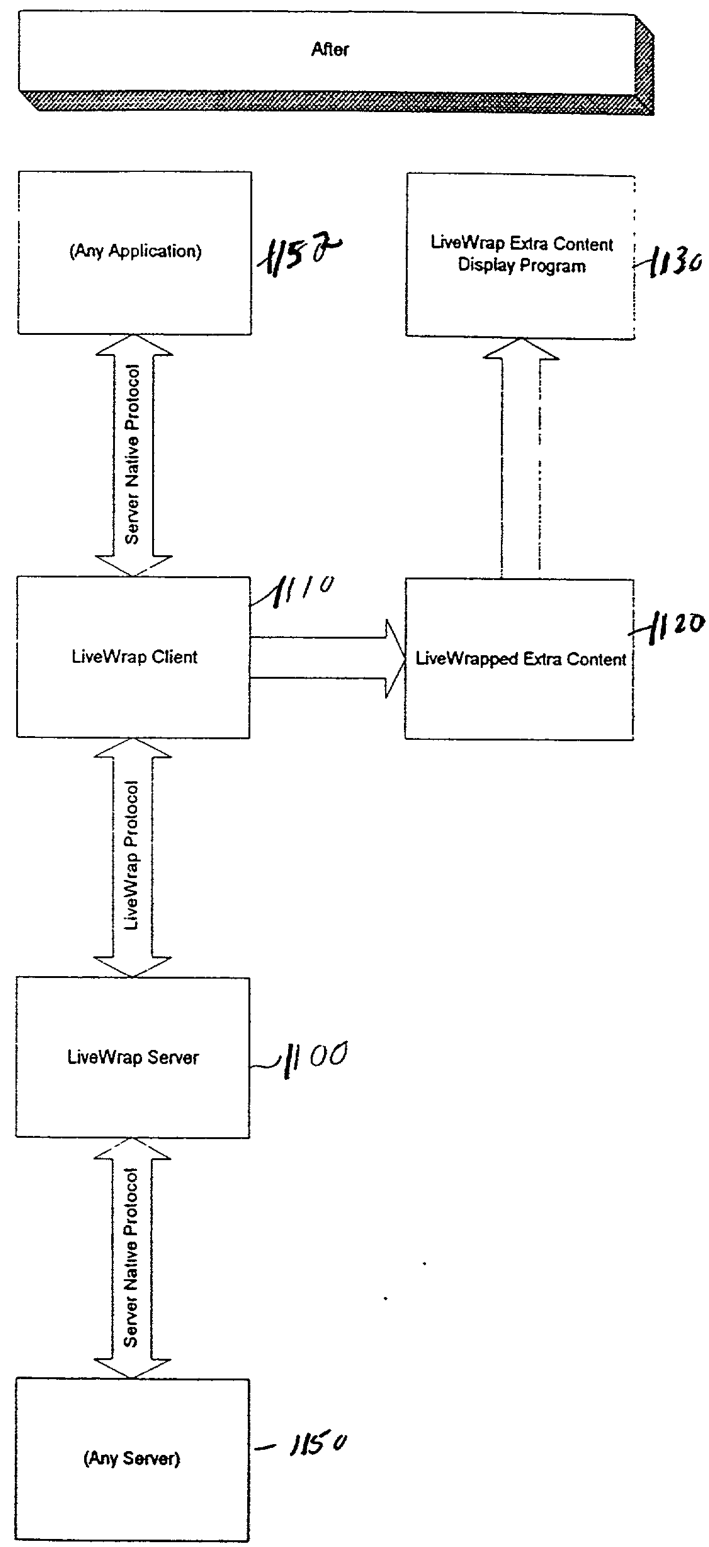


Figure 13

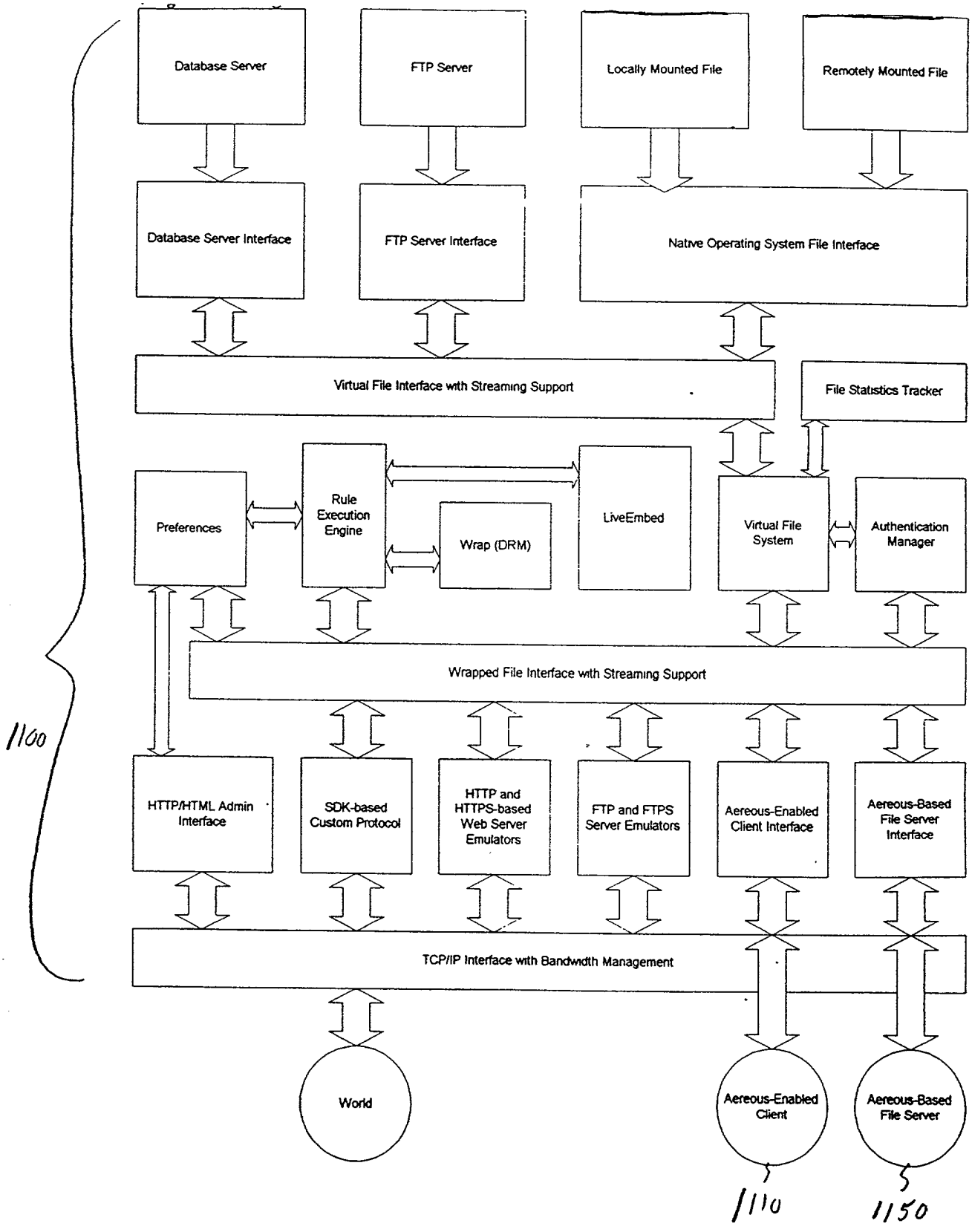

Database Server
FTP Server
Locally Mounted File
Remotely Mounted File
Database Server Interface
FTP Server Interface
Native Operating System File Interface
Virtual File Interface with Streaming Support
File Statistics Tracker
Preferences
Rule Execution Engine
Wrap (DRM)
LiveEmbed
Virtual File System
Authentication Manager
Wrapped File Interface with Streaming Support
HTTP/HTML Admin Interface
SDK-based Custom Protocol
HTTP and HTTPS-based Web Server Emulators
FTP and FTPS Server Emulators
Aereous-Enabled Client Interface
Aereous-Based File Server Interface
TCP/IP Interface with Bandwidth Management
World
Aereous-Enabled Client
Aereous-Based File Server
1100
1110
1150

Figure 14

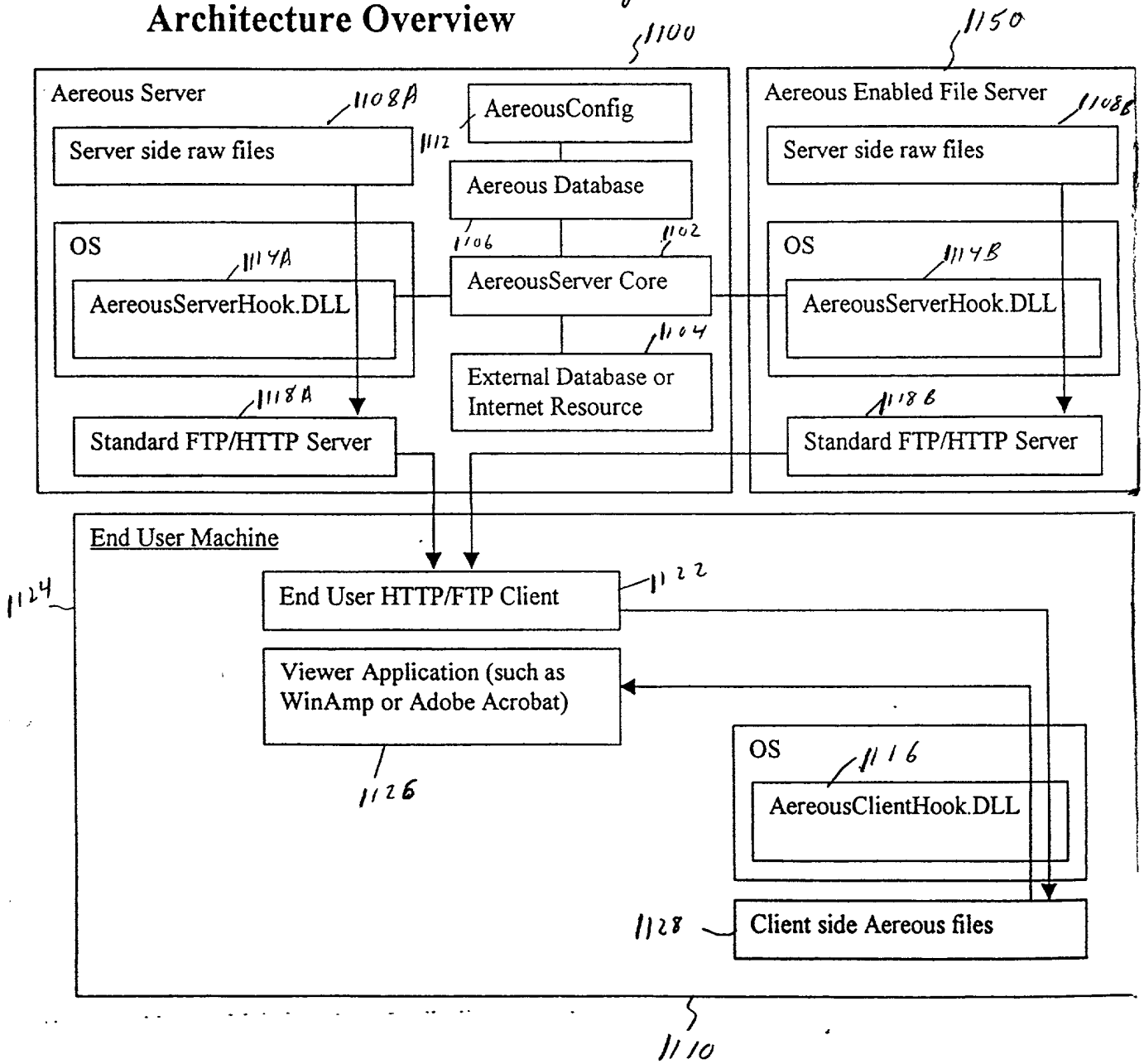

Architecture Overview
1100
1150
Aereous Server
1108A
Server side raw files
1112
AereousConfig
Aereous Database
1106
1102
OS
1114A
AereousServerHook.DLL
AereousServer Core
1104
External Database or Internet Resource
1118A
Standard FTP/HTTP Server
Aereous Enabled File Server
1108B
Server side raw files
OS
1114B
AereousServerHook.DLL
1118B
Standard FTP/HTTP Server
End User Machine
1124
End User HTTP/FTP Client
1122
Viewer Application (such as WinAmp or Adobe Acrobat)
1126
OS
1116
AereousClientHook.DLL
1128
Client side Aereous files
1110

Figure 15

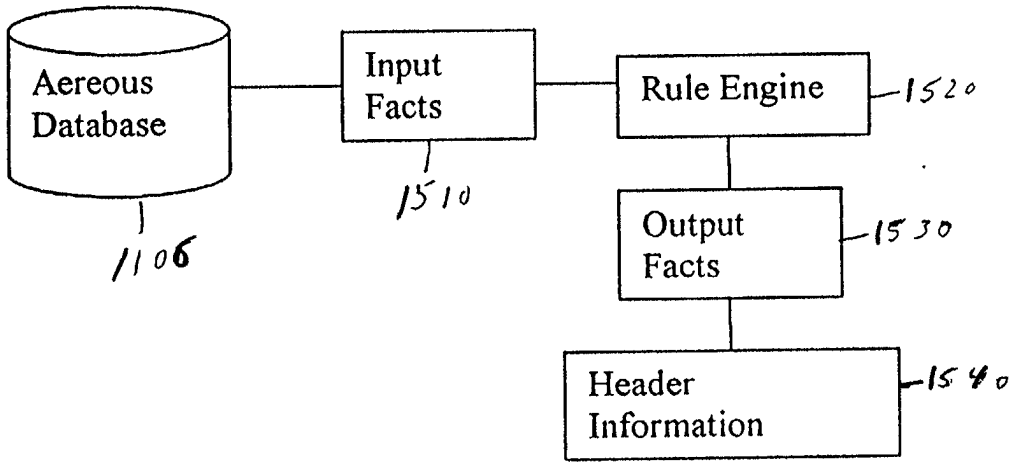


Figure16

AereousServer Core Plug-in Architecture

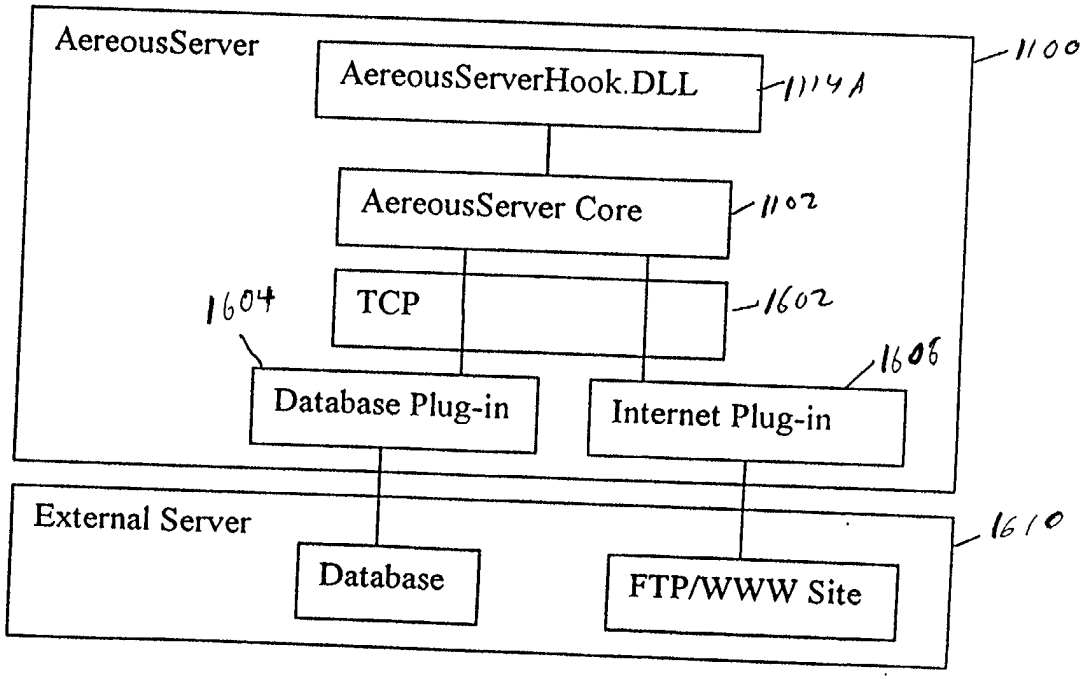


Figure 17

File Server Example Interaction Diagram

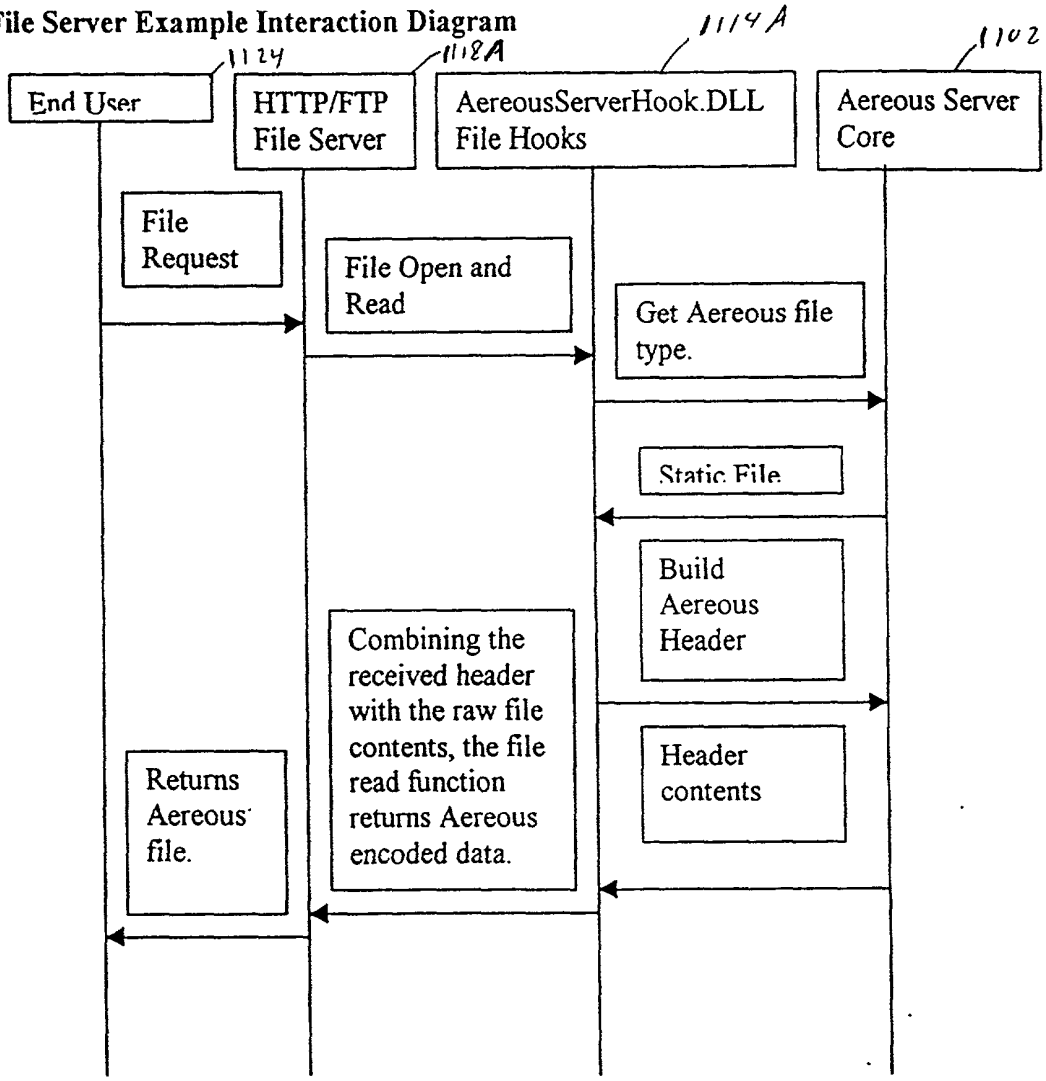


Figure 18

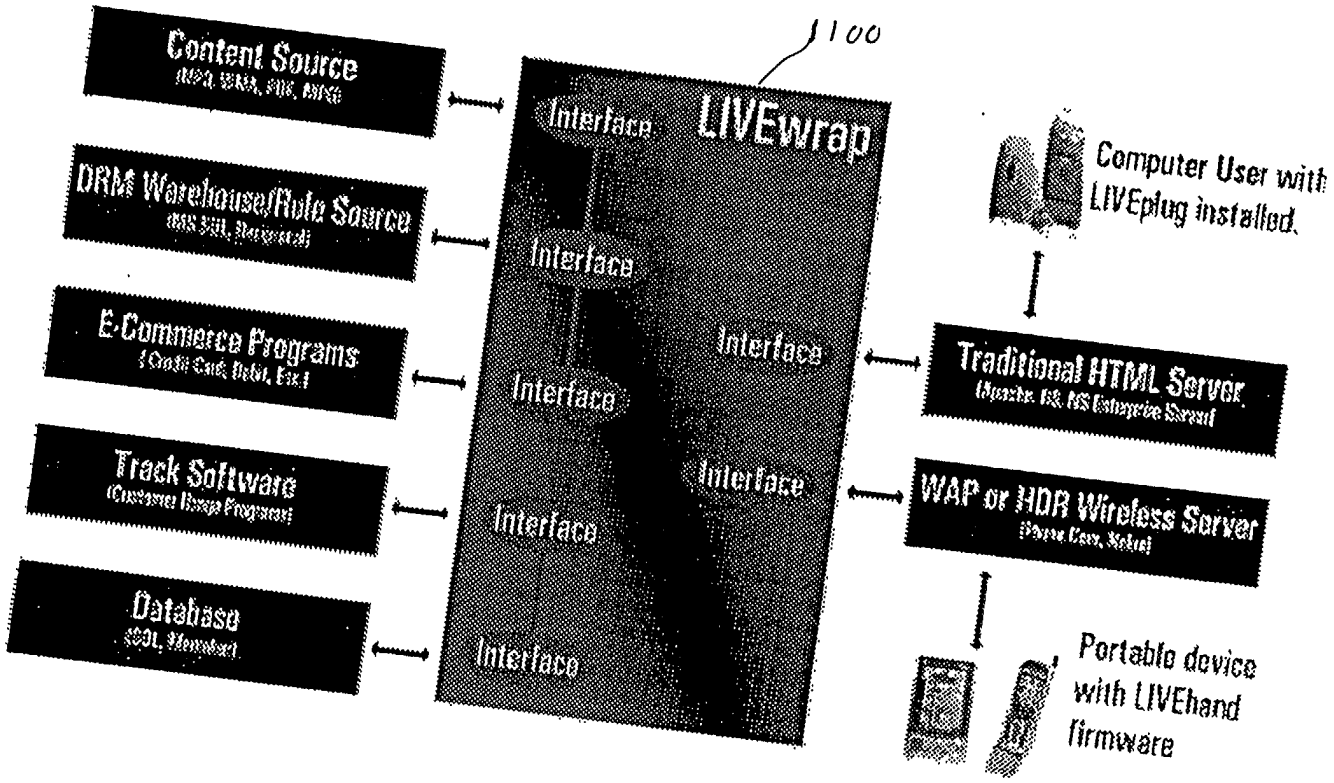

1100
Content Source
DRM Warehouse/Rule Source
E-Commerce Programs
Track Software
Database
Interface
LIVEwrap
Interface
Interface
Interface
Interface
Interface
Interface
Computer User with LIVEplug installed.
Traditional HTML Server
WAP or HDR Wireless Server
Portable device with LIVEhand firmware

Figure 19

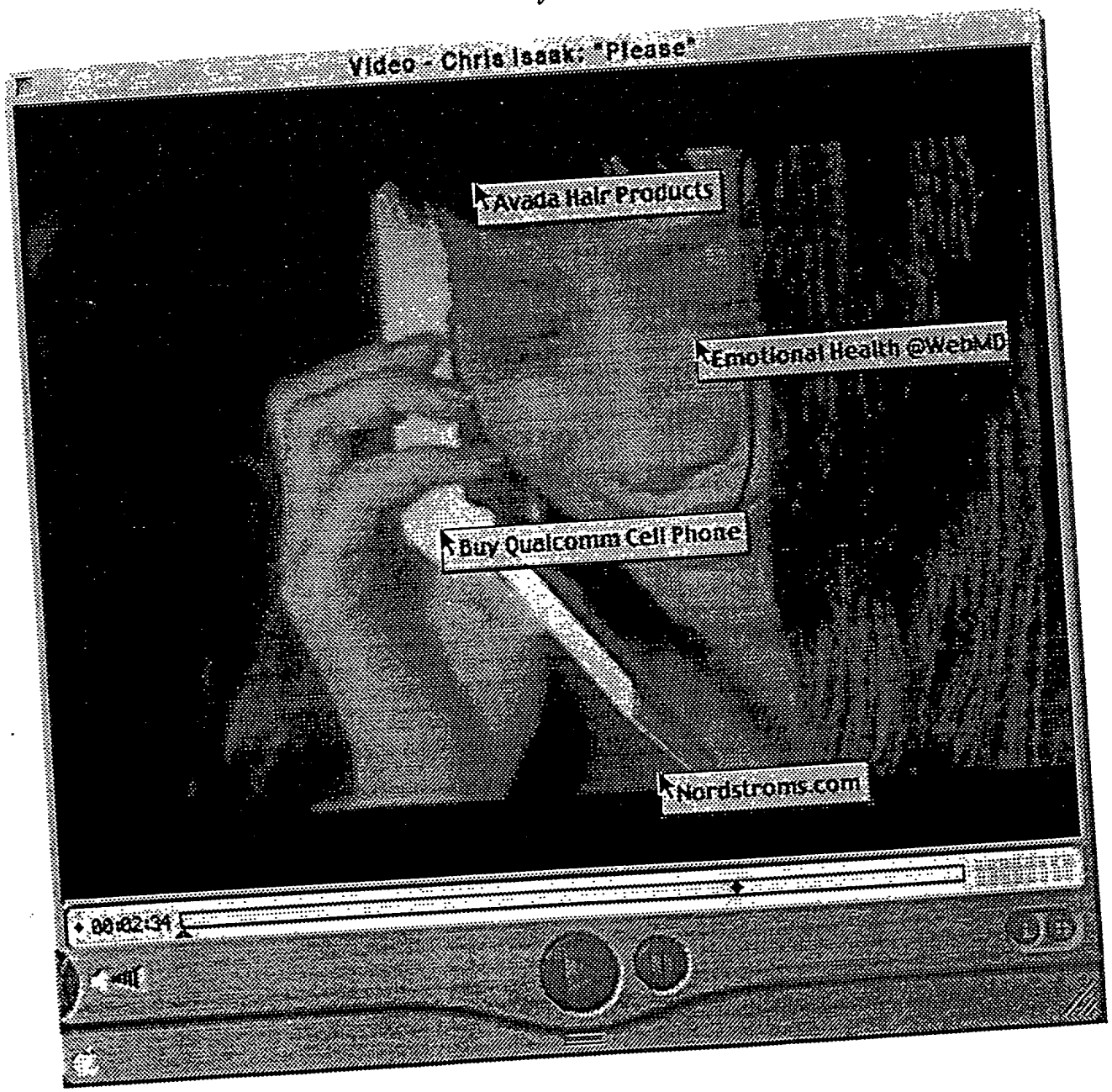


Figure 20

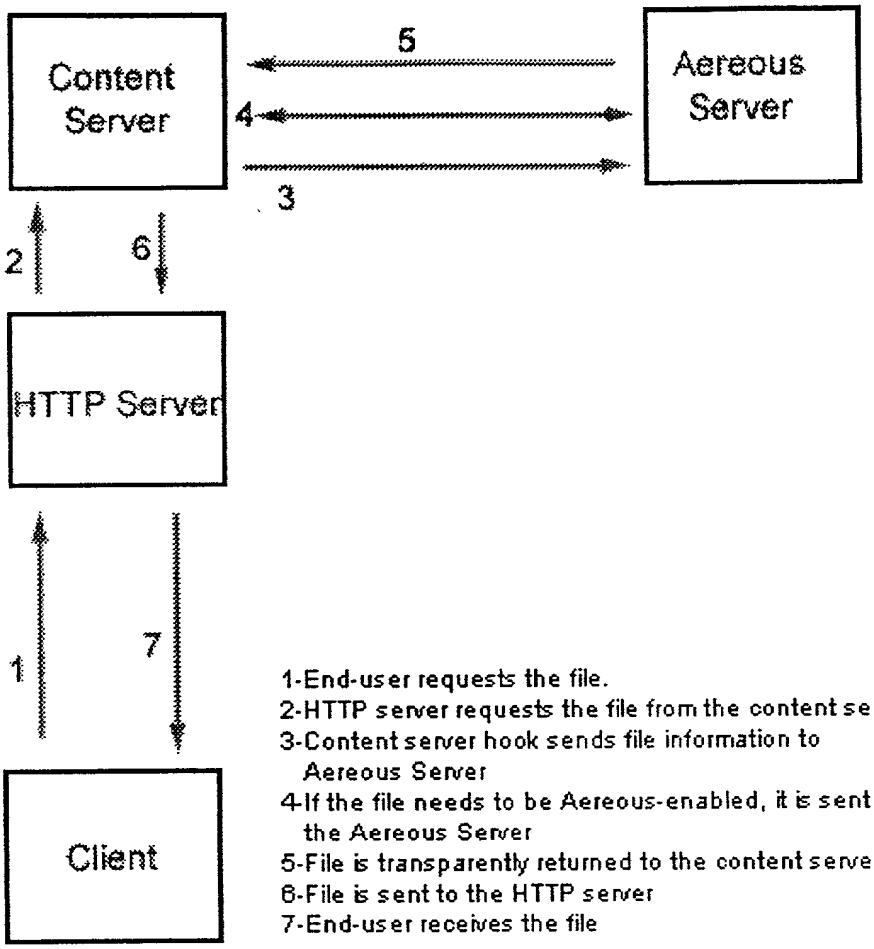
Start: 7/30/00 11:49:07 PM
PING from: 24.160.215.100
to: www. .com Every 500 mSecs

PKT#	RESULT	TIME(ms)	LENGTH	Min	Avg	Max	Timestamp	
			40	176	176	176	7/30/00	11:49:07 PM
1	success	176	40	134	155	176	7/30/00	11:49:08 PM
2	success	134						

Packets out/in/bad/%loss = 2/2/0/0:0
Round Trip Time (ms) min/avg/max = 134/155/176
• Ping Completed 7/30/00 11:49:08 PM •

Figure 21

Description	Size (bytes)	Contents
Aereous Signature	11	'AEREOUS' + 0x01301976
Aereous File Version	1	Currently 0x1
File ID	8	File's Aereous ID.
Usage Count	2	Number of usages remaining. Set to 0xFFFF for infinite usages.
Expiration Date	4	A GMT ANSI RTL style time date stamp that indicates when this file expires.
Usage Denied Content	Varying	Once a read attempt fails due to a 0 usage count, this content is displayed to the user. The format is described below under "Content Format"
Number Of Push Content Items	2	Number of items that are pushed to the user when the file is opened.
Push Content Items	Varying	Array of push content items. The format is described below as "Push Content Item Format".
Header CRC	4	A CRC value for the preceding header bytes.
Content Size	8	The size of the unencrypted data
Encryption Type	1	0 = Unencrypted 1 = 2Fish 2-255 = undefined
Encrypted Data Offset	8	A file offset to the beginning of the encrypted data. The encrypted data uses the format described in "Encrypted Data Block".



1-End-user requests the file.
2-HTTP server requests the file from the content server
3-Content server hook sends file information to Aereous Server
4-If the file needs to be Aereous-enabled, it is sent to the Aereous Server
5-File is transparently returned to the content server
6-File is sent to the HTTP server
7-End-user receives the file

Figure 12

Figure 23

VirtualFile

The VirtualFile table lists all files in the system. Each file is associated with a Plug-In and a bundle of facts that are understandable by that Plug-In.

Column	Type	Description	Sample
*VirtualFileID	Int	System assigned ID	314
Name	Text	Name of the virtual file. This is the base name, with no parent directory names and no directory separator characters. The name is not case sensitive in the server core, but is allowed to be in the database engine.	SalesReport.doc
IsDefault	Bool	Flag indicating whether Name is actually a wildcard pattern match. Use of this flag allows directories to be setup and facts associated with them without having to database each of the files that could reside within that virtual directory.	False
VirtualDirectoryID (optional)	Int	ID of the VirtualDirectory that the file resides within. Use NULL for files that reside at the root level.	4242
PlugInName	Text	Identifies which plug-in will generate the actual file contents.	FTP
IsStatic	Bool	Flag indicating whether the file is an actual static file on the server disk or a true virtual file.	True
FactBundleID (optional)	Int	Facts for this file. These facts are considered to be "owned" by this file and will be deleted if this file entry is deleted.	4243
SharedFactBundleID (optional)	Int	Facts for this file. These facts are not "owned" by this file, instead existing as shared facts to assist with centralized administration.	12000
ShouldLogUsageEvents	Int	Flag indicating whether any access to this file should result in an access log this file. ▪ 1 indicates there should be a log generated ▪ 0 means no log should be generated ▪ NULL or –1 means that the value of this setting should be inherited from the parent	1
		directory or the DefaultShouldLogUsageEvents configurable parameter	

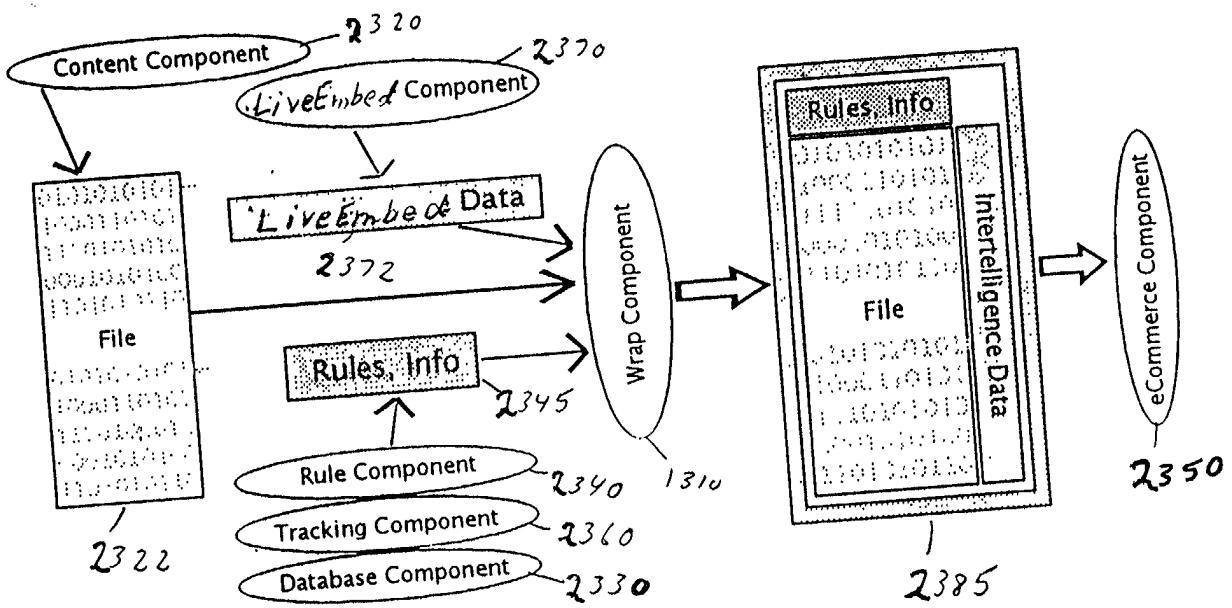
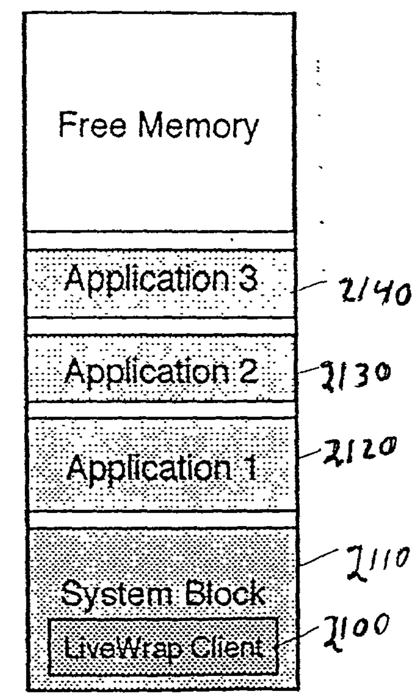

Content Component
2320
LiveEmbed Component
2370
LiveEmbed Data
2372
File
2322
Rules, Info
2345
Rule Component
2340
Tracking Component
2360
Database Component
2330
Wrap Component
1310
Rules, Info
Interelligence Data
File
2385
eCommerce Component
2350

Figure 24

Figure 25



Client Memory Space